Naval Research Laboratory
Washington, DC 20375-5320

NRL/FR/5510—92-9515

(2)

Talking to InterFIS: Adding Speech Input to a Natural Language Interface

STEPHANIE S. EVERETT, KENNETH WAUCHOPE, AND DENNIS PERZANOWSKI
Navy Center for Applied Research in Artificial Intelligence
Information Technology Division

September 11, 1992

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE September 11, 1992	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE Talking to InterFIS: Adding Speech Input to a Natural Language Interface	5. FUNDING NUMBERS PE - 62234N TA - RS34-C74-000 WU - DN2573
6. AUTHOR(S) Stephanie S. Everett, Kenneth Wauchope, and Dennis Perzanowski	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Washington, DC 20375-5320	8. PERFORMING ORGANIZATION REPORT NUMBER NRL/FR-5510—92-9515
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research Arlington, VA 22217	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report discusses the addition of speech recognition capabilities to InterFIS, the natural language interface to the Fault Isolation Shell (FIS), an expert system for troubleshooting electronics equipment. Because of the limitations of today's speech recognition technology, the addition of this capability affects the structure and flexibility of the interface; the consequences and implications of this are discussed in detail in this report. The speech recognition module is described, and a brief evaluation of system performance is presented.

14. SUBJECT TERMS Natural language processing Speech recognition Human-computer interface	15. NUMBER OF PAGES 16
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

TALKING TO INTERFIS: ADDING SPEECH INPUT TO A NATURAL LANGUAGE INTERFACE

1. BACKGROUND

InterFIS is a natural language interface to the troubleshooting module of the Fault Isolation Shell (FIS), which is an expert system development tool for the diagnosis of failures in analog electronics equipment [1]. The main functions of this FIS module are (1) to compute the probability that a particular fault hypothesis is correct after one or more tests have been performed on a particular piece of electronics equipment, and (2) to recommend the next best test based on information supplied by the diagnostician during a testing session. The original interface to FIS was standard keyboard input, where the appropriate abbreviations for all commands were displayed on the screen in a large list grouped by function [2]. A simple graphic interface was also developed, where the user invoked the commands by clicking on screen buttons labeled according to the functional grouping [3]. Later a natural language interface, InterFIS, was added [4].

InterFIS is a natural language understanding interface that accepts typed English commands as input. The PROTEUS chart parser [5] performs a syntactic analysis, producing an application-independent syntactic representation of the input sentence. This intermediate representation is mapped to domain-specific verb models by the semantic interpreter PFQAS [5] and then converted to FIS commands by the command translator COIN [4] (see Fig. 1). A natural language interface such as this is very flexible because it allows the user to paraphrase commands and to enter more than one parameter or data element at a time. For example, with InterFIS the user could type "Make a frequency test at G-IF", whereas in the other interfaces the command would be "mt" (for "Make a test"), and the parameters "frequency" and "G-IF" would need to be entered separately in response to a series of prompts.

However, using a natural language interface can be a drawback in this particular application, since typing English sentences is slow and requires the use of both hands. We determined that adding speech recognition capabilities to InterFIS would preserve the flexibility and ease of use of the natural language interface while speeding up the interaction. Using a microphone mounted in a headset would free the operator's hands, thus allowing him (or her) to manipulate equipment and to move about while performing the tests. However, speech input should augment the keyboard input, not replace it. The user should have the option of entering sentences manually in case the recognizer is having difficulty with a particular word or phrase, or when the input is not conveniently spoken, such as with a Unix file name (e.g., "/usr/fis/uut/uut-rre022").

2. SPEECH RECOGNITION COMPONENT

Hardware

The speech recognition equipment used in this implementation is the Speech Input Development System (Model DS200, Release 3.4) from Speech Systems Inc. (SSI) in Tarzana, California. It performs

Manuscript approved June 26, 1992.

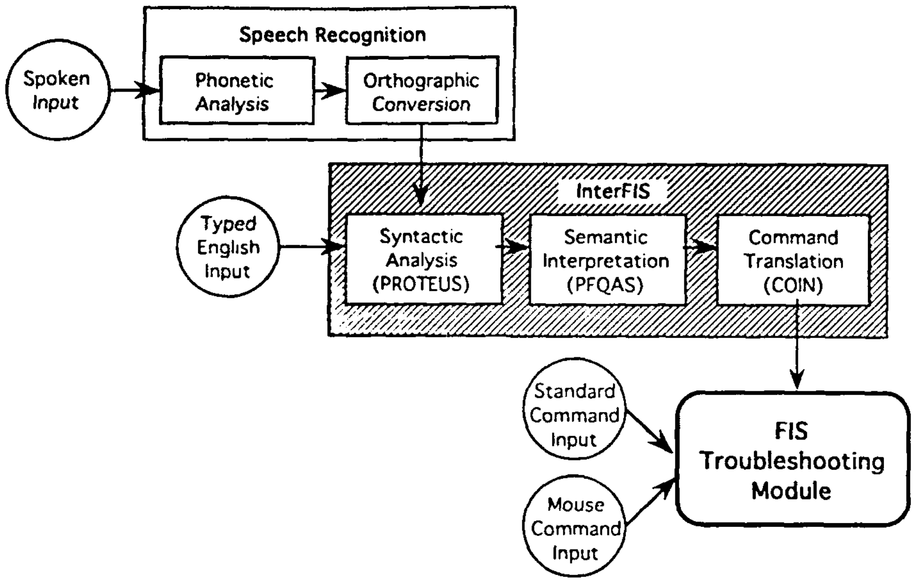

Fig. 1 — The various FIS input modes show the structure of InterFIS and the relationship of the speech recognition component to the rest of the system. The InterFIS user may enter either spoken or typed input. However, the Standard Command and Mouse Command input modes are mutually exclusive and are not supported by InterFIS.

speaker independent continuous speech recognition and is installed on a SPARCStation 2 from Sun Microsystems. The recognition hardware includes a Phonetic Engine® box that connects to the host computer's RS-232 serial port, a headset-mounted microphone, and two push-to-talk devices: one hand-held switch and one foot pedal. Input is signaled by pressing and holding either switch. The recognition software runs on the host machine, and it includes generic male and female speaker models and a phonetic pronunciation dictionary containing roughly 39,000 words.

The phonetic engine takes the live input speech and transforms it into a sequence of phonetic codes. These codes are sent to the host in real time to be translated into standard English orthography. The ASCII string produced by the SSI system is passed to InterFIS to be analyzed just as if the user had typed in the sentence at the keyboard.

Recognition Grammar

To translate the phonetic codes into standard English, the SSI system requires a grammar generated by the application developer using software tools supplied by SSI. The grammar must specify all the allowable utterances for the given application, either as an exhaustive list or as a set of context-free rules. We created a grammar containing approximately 100 rules and a vocabulary of 120 words that specifies over 100 million sentences. This grammar is very flexible, and includes numerous synonyms and paraphrases to make the system as comfortable to use as possible. For example, all the following sentences map to the FIS command SHOW-HISTORY:

Show the history list.
Display the history for me.
Give me the list of tests on the graphics monitor.
List all of the tests that were run.
What is the history?
What tests have been performed?
What are the tests that you made?

Figure 2 shows the rules used to generate these sentences.

```
SENTENCE ---> IMPERATIVE | QUESTION
IMPERATIVE ---> SHOWME (me) SHOW_NP (SCREEN_NP) |
                SHOWIT SHOW_NP (SCREEN_NP) (for me)
QUESTION ---> what IS_ARE the (current) LIST_NP |
              what tests BE CONDUCTED
IS_ARE == is are
SHOWME == show give
SHOWIT == display list show give
SHOW_NP --->    {the (current) LIST_NP} |
                {the (list of ) TESTS_NP} |
                {all (of) the {lists | {tests (REL_CLAUSE)}}}
LIST_NP ---> LISTTYPE (and LISTTYPE) {list | lists}
LISTTYPE == history probability active ambiguity fault
TESTS_NP ---> {(performed) tests} | {tests REL_CLAUSE}
REL_CLAUSE ---> REL_CONJ {{BE CONDUCTED} |
                          {you (have) CONDUCTED}}
REL_CONJ == that which
BE ---> were | {can be} | {have been}
CONDUCTED == made run done performed
SCREEN_NP ---> {on | at} the DEVICE
DEVICE == screen terminal monitor console
```

Fig. 2 — Sample rules from the InterFIS recognition grammar. These are the rules used to generate the sample sentences discussed above (and others). Capital letters indicate nonterminal symbols (categories); lower case letters indicate terminal symbols (output words). An arrow specifies a production rule; equal signs specify a list of alternative words. Vertical bars separate alternative constructions or symbols; parentheses indicate optional items. Braces enclose sequences of symbols that act as a unit.

Our recognition grammar also allows two imperatives or two questions to be conjoined into one utterance:

Make a best test and show all the lists.
What tests have been run and what is the total cost?

and allows binary compound noun phrases:

Show the ambiguity and probability lists.

This gives the operator considerable flexibility in utterance format and helps eliminate the repetition of commands. Although InterFIS is capable of processing and understanding much more complex conjoined sentences, we have restricted the speech recognition grammar to only those patterns illustrated above. Even this limited conjoining produces over 2 billion sentences; with unlimited conjoining the number of possible utterances is far too large, causing an acceptably high recognition error rate.

The flexibility of the speech recognition grammar does allow some rather unnatural utterances, such as:

What are the history?
Display the tests that was run.
Redraw a new unit.
Repeat the next test on parameters. '

Though these sentences are not apt to be spoken by the user, it is possible that the recognizer could incorrectly identify an utterance and pass an illformed sentence to InterFIS. In many cases, InterFIS is capable of handling the illformed input without difficulty. For instance, the first two examples above are accepted and executed because subject-verb agreement is not significant in this application. The last two examples can be parsed by InterFIS, but they cannot be mapped to actual FIS commands, so the interface returns error messages stating that it "cannot redraw a unit of type NEW" or "cannot repeat a test of type NEXT". We choose to allow the production of unnatural sentences by the recognizer and rely on the more powerful syntactic and semantic analyses of InterFIS to screen out unacceptable input. This makes the recognition grammar easier to write, and makes the interface more forgiving, since certain "errors", such as incorrect subject-verb agreement, do not affect the system's performance.

Responses to Prompts

When the user enters a command such as "Make a test" that requires additional information before it can be executed, the system goes briefly into a menu-based mode (using the prompts from the original FIS interface) and asks the user to enter the various test parameters. To maintain the hands-off character of InterFIS at this stage, we extended the menu functions to accept either typed or spoken input. Since the user's responses to the menu prompts are not full natural language utterances but just isolated vocalized tokens, they are passed directly to the menu functions without any natural language processing. For the same reason, the speech component does not use its main natural language grammar to recognize these utterances, but a much smaller subsidiary grammar consisting simply of a list of the 27 individual words the user can enter when in this mode. Separating the recognition process in this way increases performance, especially on the smaller vocabulary, and reflects a natural division in the task.

Connecting the Speech Module to InterFIS

The speech system includes a library of C functions (the Phonetic Decoder Interface, or PDI) for interfacing user applications to the speech module, and also provides a sample C program illustrating the use of these functions. The sample program required only minor modifications to provide us with a set of six functions for initializing the speech system, loading a speaker model, opening a grammar and dictionary, selecting a grammar, and reading an ASCII string from either the phonetic decoder or the keyboard. Since InterFIS (including the PROTEUS sentence reader) is written in LISP, these six C functions were linked into InterFIS by using the Sun Common LISP foreign function interface facility. Finally, the PROTEUS sentence reader was modified to obtain its input string by invoking the phonetic decoder/keyboard reader function instead of reading from the standard input.

3. SYSTEM PERFORMANCE EVALUATION

A small-scale test was conducted to assess the performance accuracy of the spoken interface in this domain. Using software tools included as part of the SSI system, a set of 50 random sentences was generated from the recognition grammar. Subjects (7 males, 3 females) read the sentences as they were displayed on the computer screen. They were encouraged to familiarize themselves with each sentence before speaking, and to speak in a "natural" way. They were instructed in the use of the push-to-talk button, but were given no other training or practice. This test evaluated the speech recognition component only; it did not include any natural language processing and did not provide any feedback to the subject.

Figure 3 shows the test results. Each utterance is treated as a single entity, and as such it is either all right or all wrong. The raw accuracy score is the basic utterance recognition score: if the string produced by the speech recognition system is not identical to the prompt text, that utterance is wrong.

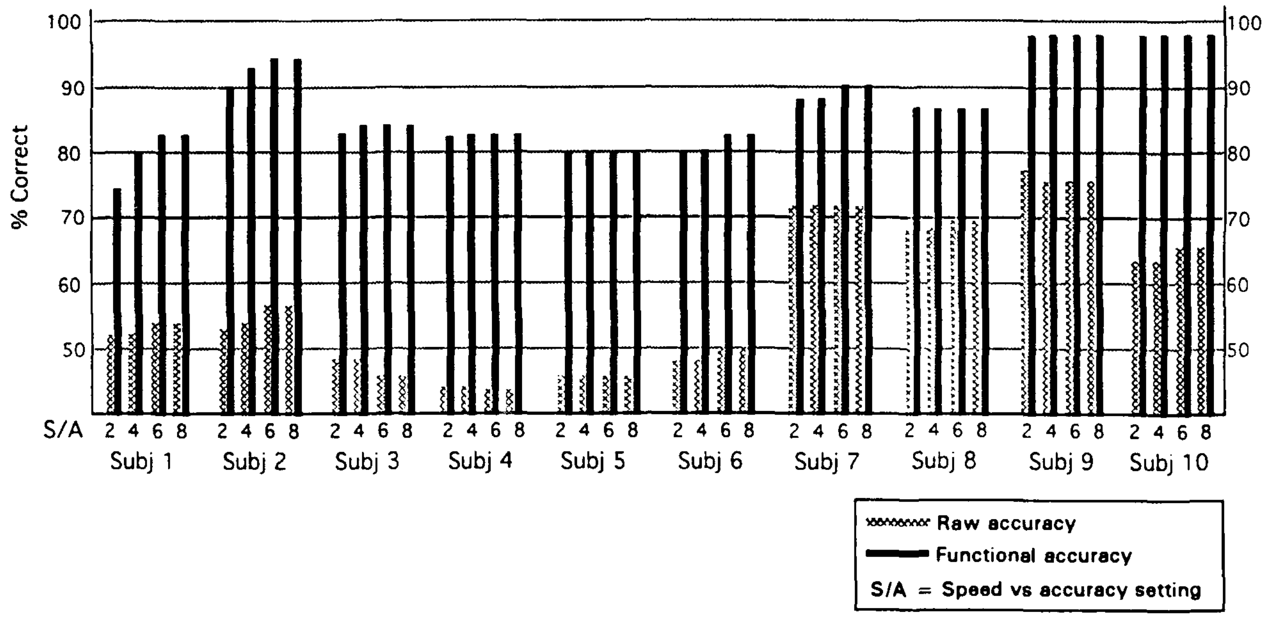

Fig. 3 — Recognition accuracy scores (% correct) for a set of 50 sentences. Raw accuracy is the percent of utterances correctly recognized; functional accuracy is the percent of utterances that would have elicited the desired action or response from the expert system even if there were errors in the recognition.

For the functional accuracy score, a given utterance is counted as correct if the string produced by the speech recognition system would have resulted in FIS performing the desired action, even if the string does not match the prompt. Only those strings that would have resulted in error messages or undesired actions are counted as errors. We feel that this second score is more indicative of the actual usability of the system.

The speed vs accuracy setting shown in Fig. 3 is a variable recognition parameter that controls the trade-off between processing speed and recognition accuracy for the SSI system. The settings can range from 0 to 9, with 0 being the fastest but with the least attention to accuracy, and 9 being the slowest but having the greatest potential accuracy. The four settings tested in this evaluation were chosen to give a profile of the trade-off curve for this grammar. The results indicate that the speed vs accuracy setting has no significant effect on recognition performance for this implementation ($t = 0.15$ for raw accuracy, $t = 0.26$ for functional accuracy; $p > 0.05$). A setting of 6 was sufficient to realize maximum performance for all subjects. On the Sun Sparc2 used for this test all the utterances were processed in less than 50% real time, meaning that a 1 s utterance would be decoded in less than 500 ms (the processing time ratio ranged from 0.23 to 0.36). On a slower machine, the phonetic decoding speed would probably be more of a consideration in determining the optimal speed vs accuracy setting, but in this situation we can choose based on accuracy alone.

As can be seen from Fig. 3, the functional accuracy of the spoken interface is considerably higher than the raw recognition accuracy. Functional accuracy averages 85.8% correct, even though the raw accuracy averages only 57.2% correct. The difference in the scores is attributable to the structure of the speech recognition grammar, and the fact that the recognizer always generates an acceptable command as defined by that grammar. The flexibility of the grammar and the ability to paraphrase commands introduce redundancy, so unless the recognizer makes major errors, the generated command will usually produce the desired action.

Subjects 1 through 8 had had no formal speech training and little or no experience speaking into microphones (except telephones). The variability in their scores, especially the raw recognition accuracy scores, is consistent with our observations of their speech habits: subjects 3, 4, and 5 all spoke quickly and tended to mumble; subjects 7 and 8 spoke much more slowly and carefully. Subject 6 spoke fairly carefully but had a moderate Hispanic accent. It is interesting to note that this accent did not have a noticeable effect on the recognition scores. Subjects 9 and 10 had both had speech training and considerable experience with microphones, including speech recognition systems. Their scores indicate that training and practice can dramatically improve recognizer performance. The performance of the other subjects would probably have been better if they had been instructed to speak slowly and carefully, and they would also be expected to improve with practice.

Another way to improve accuracy scores might be to introduce an utterance acceptance threshold. For testing purposes no threshold was used; all utterances were accepted and processed, and the closest match determined. With an acceptance threshold, utterances scoring below a certain confidence level would be rejected, and the user would be asked to repeat the command. This could help eliminate incorrect responses, especially if the user misspoke, or if extraneous noise interfered with the recognition of the command. Using analysis tools provided by SSI, it was determined that an utterance acceptance threshold of approximately 700 (available range: 0 to 1000) would be suitable for this application. Optimum threshold values for individual speakers ranged from 675 to 745, with a mean of 712 and a standard deviation of 24.7.

Scores might also improve with the use of customized speaker models as opposed to the generic male and female speaker models (included with the SSI system) that were used in this test. However, creating a customized model requires several hours, and it is recommended only if the speaker is having difficulties with the recognizer. For military applications it is highly preferable to use generic speaker models, because it is not always possible to take the time for a new user to train the system. The generic models performed quite satisfactorily for all the speakers in this test.

On the whole, the addition of speech recognition capabilities has significantly enhanced the "friendliness" and ease of use of the FIS system. All the researchers have found that they much prefer using spoken input to using typed input. Spoken interaction is faster, especially compared to typing out English commands. Speech is a comfortable, natural input mode, and frees the user's hands. Though the recognition accuracy is not perfect, it rarely interferes with the operation of the system, at least for practiced speakers. From our experience, users of InterFIS seem quite tolerant of occasional errors, even when the action performed differs significantly from what was requested. Only when the system repeatedly fails to understand a particular utterance do the users begin to get frustrated. After about three failures, the users usually resort to keyboard entry of the command and then return to speech for subsequent commands.

4. GRAMMAR OPTIMIZATION

The grammar originally developed for InterFIS (here called "Grammar I" for reference) had broad grammatical coverage and extensive natural language processing (NLP) capabilities, but no effort had been made to optimize its response time. The interface was powerful, but seemed unusually slow for such a restricted domain and task, and felt uncomfortably sluggish, particularly when the comparably slow typing of sentences at the keyboard was replaced by the rapidity of speech input. To make the speech and natural language interface a desirable alternative to other interface modalities, we had to significantly improve the system's response time by making the syntactic grammars more efficient.

Our syntactic grammars consist of context-free production rules augmented by syntactic constraint rules called restrictions. The parsing algorithm is a form of generate and test: the production rules generate candidate analyses and the restrictions apply tests to accept or reject the candidates. A number of possible sources of inefficiency exist in such a grammar. There could be more productions than needed (overcoverage), resulting in too large a search space; not enough restrictions (underproscription), resulting in an inefficiently pruned search space; or more restrictions than needed (overproscription), spending excessive time performing unnecessary tests. Particular rules might be firing an inordinate amount of the time, thus contributing more to the problem than other rules.

However, the addition of the speech component to the front end of the system meant that its highly constrained grammar could be treated as a syntactic/semantic filter that would make much of the search and constraint checking by our present syntactic grammar unnecessary. For example, if the phonetic recognizer mistakenly heard the user say "What are the history" instead of "What is the history", the intended meaning is still clear and we would not want an overly restrictive grammar to penalize the user with a "Sorry, don't understand" response. Hence our objective was to determine which features of Grammar I were primarily responsible for its inefficiency, develop a more efficient and pruned grammar still capable of processing all the sentences passed to it by the speech component, and see if the result showed an acceptable response time.

Grammar I

Grammar I consists of 155 phrase structure rules, 73 syntactic constraint rules or restrictions, and one conjunction "metarule" that adds compounded versions to fifteen of the phrase structure rules. Twenty-two of the 73 syntactic constraints are conjunction restrictions. When tested on a corpus of 52 sample sentences averaging 7 words in length, Grammar I yielded a mean parsing time of 8.3 s per sentence, ranging from a best case of 3.3 s to a worst case of 23.0 s (these tests were run on a Sun 3/260 workstation). We found that 35% of the time spent in the test phase of the generate-and-test was to constrain occurrences of the empty category WH (which marks the trace of a moved noun phrase) to occur only in the context of WH-questions ("Which unit[i] did you load WH[i]?") and relative clauses ("The unit[i] that you loaded WH[i]"). Another 32% of the time was spent applying proscriptive restrictions like subject-verb agreement, many of which, though linguistically appropriate, are not really necessary for ambiguity pruning in the InterFIS domain.

The remaining 33% of the testing time was spent running conjunction restrictions, even though there were only five compounds in the entire corpus of 52 sentences. Since these restrictions only run when a compound phrase structure analysis is being considered, it was apparent that overgeneration of candidate compounds was the primary source of inefficiency. More specifically, fourteen of the 22 conjunction restrictions handle a specialized construction called a CONJOMIT in which an item to the left or right of a conjunction is omitted, as in THE ACTIVE [LISTS] AND HISTORY LISTS. Of the eight nonterminals Grammar I allowed to be omitted in this way, only one was considered necessary for incorporation into the speech recognizer, making the remaining CONJOMIT generation and testing by the grammar superfluous.

Grammar II

The new grammar we developed for speech-input InterFIS, Grammar II, has half as many phrase structure rules (81), half the number of empty categories (3), one-fifth as many compounded rules (3), one-seventh the number of restrictions (10), and one-tenth the number of conjunction restrictions (2). On the test corpus it ran an average of 5.7 times faster than Grammar I, with a mean parsing time of 1.5 s per sentence, a best case of 0.5 s and a worst case of 4.1 s. The worst case for Grammar II was thus less than one second slower than the best case for Grammar I. The average response time of 1.5 s for Grammar II felt quite acceptable in the interactive environment.

Table 1 — Illustrations of Compounding Permitted in InterFIS

Production	Illustration
Question	*what units did you load and what did you show?*
Imperative	*load a unit and make a test.*
Noun phrase	*the active list and the history list*
Premodifier	*a new, unloaded unit*
Assertion	*a unit was loaded and a test was made.*
Passive agent	*was a unit loaded by you and FIS?*
Prepositional phrase	*draw the unit on the screen and on the display.*
Past tense VP*	*have you shown a unit and shown the lists?*
Passive VP*	*was I shown a unit and shown the lists?*
Progressive VP*	*are you loading a unit and showing the lists?*
Untensed VP*	*will you load a unit and show the lists?*
Relative clause	*the unit that you loaded and that you drew*
WH question	*when did you load it and why?*
Yes-No question	*did you load a unit and did you make a test?*

*VP: verb phrase

Table 2 — Execution Statistics for Grammar Optimization

	Grammar I		Grammar II		Improvement	
	num	s	num	s	num	s
Edges Generated						
Active	40883	61.7	15633	20.4	2.6x	3.0x
Inactive	7487	44.7	4791	19.3	1.6x	2.3x
Total	48370	106.4	20424	39.7	2.4x	2.7x
Restrictions Fired						
Conjunction	15232	139.8	458	5.0	33.3x	28.0x
Null WH	12017	146.8	3941	35.5	3.0x	4.1x
Other	13120	137.6	754	13.3	17.4x	10.3x
Total	40369	424.2	5153	53.8	7.8x	7.9x
Total Time		530.6		93.5		5.7x

Number of sentences: 52
Total number of words: 359
Number of conjoinings: 6 (5 sentences)

Grammar II spent a full 66% of its time on the highly context-sensitive job of WH-pruning, and 25% of its time on other miscellaneous restrictions. It thus spent only 9% of its test phase on conjunction restrictions (vs 33% by Grammar I), much more in line with the 11% of the test sentences that actually contained compounds. Table 1 lists the fifteen different types of compound admitted by Grammar I, the first three of which were retained in Grammar II, corresponding as they do to compounds recognized by the speech component. Grammar II allows the CONJOMIT construction in both imperatives (e.g., "Load (a new unit) and display a new unit") and noun phrases ("The history (list) and ambiguity lists"), although the speech component recognizes such omissions only in specific noun phrases. Table 2 shows an overall comparison between the two grammars. The "Edges Generated" section represents the generate phase of chart parsing, where Active edges are incomplete constituents searching for their remaining children, and Inactive edges are complete well-formed substring analyses. The "Restrictions Fired" are the tests that are applied whenever particular types of active or inactive edges are generated.

Observations

Broad coverage grammars consisting of highly generalized phrase structure rules constrained by restrictions can rapidly lead to overgeneration and overtesting. If that happens, then one should first remove all grammar productions that are not relevant to the present domain, and keep the grammar to the minimum size necessary and sufficient to correctly handle the anticipated input data, adding rules only as they become necessary. In the rules that remain, one can sometimes transfer some of the "test" work into the "generate" phase by encoding the production rules in a more constrained manner. For example, Grammar I's treatment of compound questions was along the following lines:

<QUESTION> ::= <QUESTION> <CONJWORD> <QUESTION>.[1]

RESTRICTION: the first daughter of a compound may not itself
be a compound.

Here it is the restriction that enforces right-branching recursion only. In Grammar II the treatment is as follows:

<QUESTIONS> ::= <QUESTION> | <QUESTION> "AND" <QUESTIONS>.[1]

The restriction is no longer needed because the production rule itself enforces the right-branching-only constraint; the parser is steered top-down in the right direction during generate phase rather than overgenerating bottom-up followed by pruning. This approach is not as modular as the overgeneralize/restrict approach, introducing as it does nonterminals like QUESTIONS that are not bona fide phrase markers in the String Grammar [6] formalism to which we try to adhere. But since the speech recognizer serving as the front end of the system enforces these constraints in a similar manner, it is not necessary for subsequent links in the processing chain to remain more generalized than that first link, a topic to be explored further below.

5. DISCUSSION

Ease of Representation

Despite the convenience of working with an accurate, off-the-shelf speech recognition product accompanied by a variety of useful development software, the necessity of providing the SSI speech system with an independent grammar written in a relatively weak formalism (limited-recursive context free rules) made it awkward to integrate into our existing NLP system.

[1]These rules are in the InterFIS syntactic analysis grammar (PROTEUS). They are given in Backus-Naur Form (BNF) notation, not in the SSI notation used for the recognition grammar in Fig. 2.

Although we could translate the context-free portion of the NLP syntactic grammar directly into SSI notation with little difficulty, we soon found that a simple phrase structure grammar was too unconstrained to yield an acceptable speech recognition rate. To translate the syntactic restrictions and semantic constraints of the NLP system into SSI notation required abandoning the phrase structure grammar approach and instead writing a semantic grammar from scratch. A semantic grammar is a grammar written in conventional context-free notation whose nonterminal symbols are domain-specific semantic word classes rather than syntactic phrase markers. Writing and maintaining such grammars is notoriously difficult compared to the modular approach used in an NLP system such as InterFIS, where each class of information (lexical information, phrase structure, co-occurrence constraints and semantic constraints) is encoded as a separate knowledge base in a notational formalism convenient and appropriate to the type of information being represented. Encoding these constraints in the much weaker notational formalism of context-free rules is awkward and time-consuming, and the necessity of coordinating and maintaining two completely different natural language grammars, one for speech recognition and the other for structural analysis, is bothersome.

Recognition Power

Another limitation of the speech module lies in its recognition power. Although the grammars for the SSI speech recognizer are written in context-free notation, they do not support arbitrary recursion. Though the grammar may contain recursive productions, it must be compiled with a flag limiting recursions to some predetermined fixed number. As a result, these grammars are equivalent to noniterative regular expression languages, or finite languages (complete enumerations of strings of predetermined length). As such they do not even have regular-expression recognition power (Type 3 in the Chomsky hierarchy), whereas the recognition power of our NLP system is at least that of a context-sensitive recognizer (Type 1) and probably higher (unrestricted or Type 0). Since the speech module sits at the front end of the system, the full capability of the NLP modules is thereby underutilized.

Recursive phrase structures like compounds and noun modifiers (adjectives, relative clauses and prepositional phrases) are an important feature of natural language. However, their use is constrained by human memory limitations, and can be artificially restricted by the particular demands of the application task. The InterFIS speech grammar does not employ recursion at all. We did incorporate binary compounding of sentences and certain noun phrases into the grammar, but these were explicitly coded, not recursive productions. In other tasks like database management, the arbitrary nesting in user input (e.g., "Show the salaries of all the managers who made more than the highest paid salesman they hired last year") might make true recursion essential.

The speech module does not adhere to the philosophy of extensibility and portability around which our NLP system has been designed. The NLP system is based on the derivation of application-independent linguistic representations (phrase structure trees and logical forms) that are then translated into an application-specific form (FIS commands). The linguistic representations are produced by general-purpose interpretive modules—a parser and semantic interpreter—that have been provided with knowledge bases (grammar, dictionary, and semantic model) appropriate to the application. In extending the system or porting to a new application, only the knowledge bases need to be altered: the grammar usually very little, the others somewhat more so. In each case, the modifications and additions to be made are simple, and their effects are highly predictable because of the logical, modular organization of the knowledge bases. As we have seen, however, the speech grammar must be made highly domain-specific to provide an acceptable accuracy rate, and semantic grammars are notorious for their lack of easy extensibility and portability. Transferring the resulting system to a new application would require completely discarding the old speech grammar and starting from scratch on a new one.

Redundancy of Effort

The NLP system is organized as a series of interpreters of decreasing output bandwidth, each applying a finer grain of discriminative knowledge than the one before it. Specifically, the lexical analyzer takes as input any string of ASCII characters but recognizes only those that correspond to sequences of English words; the parser recognizes only those English word sequences that are syntactically well formed; and the semantic interpreter accepts only those parses that are meaningful. Since the speech grammar must duplicate most of the lexical, syntactic, and semantic knowledge used by the NLP modules downstream from it, the speech module acts like a narrow bandpass filter that transmits only strings that are already lexically, syntactically, and semantically well formed. Because of the limited recognition power of the speech module, the bandwidth of data input to the NLP system is as narrow as, or even narrower than the bandwidth of data the system outputs. In effect the speech module "usurps" the role of the NLP modules as recognizers, rendering their powerful type-checking and search facilities redundant and reducing them to mere structure builders.

At present the NLP modules must still be invoked to produce the application-independent logical form for translation into a FIS command. But since it is possible to have the speech module produce parse trees of the strings it recognizes, it may be that the NLP components are not even needed for structure building in this system, and they could be eliminated altogether. Since the speech grammar is a domain-specific semantic grammar, the trees it generates might require only a small application-specific processor to be mapped to the target (FIS command) language. Although this runs contrary to the portability approach we have long advocated, we have shown that the necessity for a semantic grammar in the speech component impairs true portability anyway.

Observations

In an ideal system, the speech module would not require its own independent grammar but would interact with the other NLP software and use their knowledge bases to constrain its search. With that sort of fully interleaved approach the aforementioned problems would be avoided. (This would be similar to the "layered blackboard" approach used in some earlier research systems [7].) However, this approach requires considerably more computation than the current approach, and it is not a practical option given today's processing technology.

6. SUMMARY

Overall, the addition of speech recognition capabilities to the existing natural language interface was successful. The interface remains flexible and easy to use, and input speed has increased. The ability to speak commands into FIS frees the user's hands to manipulate tools or equipment, and with the addition of speech output he or she would be able to move about while conducting the tests.

However, the limitations of the current speech recognition technology restrict the linguistic capabilities of the natural language interface. For this particular application this is not a significant problem; the linguistic restrictions are sufficiently offset by the improved usability and speed of the interaction. For other more complex applications, the linguistic restrictions imposed by adding speech recognition might be unacceptable, especially if the existing interface includes a sophisticated natural language processing system. As speech recognition and computational technologies improve, we can expect to see significant increases in the capabilities of speech recognition interfaces. Until then, the pros and cons of adding speech recognition to an existing interface should be weighed carefully.

REFERENCES

1. Pipitone, Dejong, Spears and Marrone, *The FIS Electronics Troubleshooting Project* in *Expert Systems Applications to Telecommunications*, J. Liebowitz, ed. (Wiley and Sons, New York, 1988), pp. 73-101.

2. Pipitone, Dejong, Spears, "An Artificial Intelligence Approach to Analog Systems Diagnosis," NRL Report 9219, 1989.

3. R. Schoeffel, "FIS Ergonomic Interface Users Guide," 1988. (unpublished manuscript)

4. D. Perzanowski and B. Potter, *InterFIS: Natural Language Interfacing to an Expert System Shell*, in *Expert Systems World Congress Proceedings*, J. Liebowitz, ed. (Pergamon Press, NY, 1991), 2, pp. 1086-1094.

5. R. Grishman, "PROTEUS Parser Reference Manual," PROTEUS Project Memorandum #4, New York University, New York, 1986.

6. N. Sager, *Natural Language Information Processing* (Addison-Wesley, Reading, MA), 1981.

7. L. D. Erman, F. Hayes-Roth, V.R. Lesser and D.R. Reddy, "The Hearsay-II Speech-Understanding System: Integrating Knowlege to Resolve Uncertainty", Computing Surveys, 12(2) (June 1980).